UNIVERSITI TEKNOLOGI MARA

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES (FSKM)

FLUID MECHANICAL AND HEAT/MASS TRANSFER (MAT 720)

ASSIGNMENT 2

Prepared By	**NUR IMAN NAZIRAH BT NASIR (2018469592)** **NUR ELLIA BINTI NORAZAM (2018680058)** **NURUL AINA SYAFIQAH BT ZULKAFLI (2018632812)**
Group	**CS7731B**
Program	**CS773**
Faculty	**FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES**
Prepared for	**DR. SERIPAH BT AWANG KECHIL**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

A partial differential equation (PDE) is any equation that involves a function of more than one independent variable and at least one partial derivative of the function. The PDE's order is the highest order derivative that appears in the PDE. The principal part of a PDE is the collection of terms in the PDE containing derivatives of order equal to the order of the PDE. Some of the examples are

heat equation:

$$c\frac{\delta^2 t}{\partial x^2} = \frac{\partial u}{\partial t}$$

and wave equation:

$$c^2\frac{\partial^2 t}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}$$

PDE can be solve using analytical methods and numerical methods. One of the differences of these methods is that analytical methods give exact solutions, meanwhile numerical methods give approximate solutions. In this assignment, we will be focusing on Finite-Difference Method and Crank-Nicholson Method under heat equation.

The Finite-Difference Method is one of several techniques for obtaining numerical solutions to solve the heat equation. In all numerical solutions the continuous partial differential equation (PDE) is replaced with a discrete approximation. In this context the word "discrete" means that the numerical solution is known only at a finite number of points in the physical domain. The number of those points can be selected by the user of the numerical method. In general, increasing the number of points not only increases the resolution (i.e., detail), but also the accuracy of the numerical solution.

The Crank–Nicholson Method is a finite difference method used for numerically solving the heat equation and similar partial differential equations. It is a second-order method in time. It is implicit in time and can be written as an implicit Runge–Kutta method, and it is numerically stable. The method was developed by John Crank and Phyllis Nicolson in the mid-20th century(Nicotson, 1947). For diffusion equations (and many other equations), it can be shown the Crank–Nicholson Method is unconditionally stable. However, the approximate solutions can still contain (decaying) spurious oscillations if the ratio of time step Δt times the thermal diffusivity to the square of space step, Δx 2, is large (typically larger than 1/2 per

Von Neumann stability analysis). For this reason, whenever large time steps or high spatial resolution is necessary, the less accurate backward Euler method is often used, which is both stable and immune to oscillations.

OBJECTIVES

The objectives of this project are:

1. To study the result of different type of Finite Difference Method and Crank-Nicholson Method.
2. To study the relationship between Finite Difference Method and Crank-Nicholson Method.
3. Be able to differentiate between Finite-Difference Method and Crank-Nicholson Method in solving PDEs.
4. Be able to solve Finite-Difference Method and Crank-Nicholson Method in Heat Equation PDEs using MATLAB software.

METHODOLOGY

For one-dimensional heat equation we can substantiate by using Finite-Difference Method and Crank-Nicholson Method.

In **Finite-Difference Method**, the standard partial-differential equation is

$$\alpha^2 \frac{d^2u}{dx^2}(x,t) = \frac{du}{dt}(x,t) \quad for\ 0 < x < 1\ and\ t > 0 \tag{1}$$

With boundary conditions

$u(0,t) = u(1,t) = 0 \quad for\ t > 0$

$u(x,0) = f(x) \qquad for\ 0 \le x \le 1$

By using the central difference and the forward difference approximation

$$\frac{d^2u}{dx^2}(x,t) = \frac{u(x_{i+1},t_j) - 2u(x_i,t_j) + u(x_{i-1},t_j)}{h^2} \tag{2}$$

$$\frac{du}{dt}(x,t) = \frac{1}{k}[u(x_i,t_{j+1}) - u(x_i,t_j)] \tag{3}$$

Then we substitute equation (2) and (3) into (1),

$$\frac{w_{i,j+1} - w_{i,j}}{k} - \alpha^2 \left[\frac{w_{i+1,j} - 2\, w_{i,j} + w_{i-1,j}}{h^2}\right] = 0 \tag{4}$$

We simplifying the equation (4),

$$w_{i,j+1} = \left(1 - \frac{2\alpha^2 k}{h^2}\right) w_{i,j} + \frac{\alpha^2 k}{h^2}\left(w_{i+1,j} + w_{i-1,j}\right) \tag{5}$$

Now we let $\lambda = \frac{\alpha^2 k}{h^2}$

$$w_{i,j+1} = (1 - 2\lambda)\, w_{i,j} + \lambda\left(w_{i+1,j} + w_{i-1,j}\right) \tag{6}$$

In general, if we use the equation (6) to determine value w on the (j+1), we need to solve a linear system AX=B, where the coefficient matrix A is tridiagonal matrix.

$$A = \begin{bmatrix} (1-2\lambda) & \lambda & 0 & 0 \\ \lambda & (1-2\lambda) & \lambda & 0 \\ 0 & \lambda & (1-2\lambda) & \lambda \\ 0 & 0 & \lambda & (1-2\lambda) \end{bmatrix}$$

Then the approximate solution is given by: $\quad w^{(j+1)} = Aw^{(j)}$

In **Crank-Nicholson Method**, we used the averaged difference method;

$$\frac{w_{i,j+1}-w_{i,j}}{k} - \frac{\alpha^2}{2}\left[\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2} + \frac{w_{i+1,j+1}-2w_{i,j_1}+w_{i-1,j+1}}{h^2}\right] = 0$$

This method also can be expressed in the matrix form;

$$Aw^{(j+1)} = Bw^{(j)},\ \text{for each } j = 0,1,2,\ \ldots$$

where

$$\lambda = \alpha^2 \frac{k}{h^2},\ w^{(j)} = (w_{1j}, w_{2j}, \ldots, w_{m-1,j})^i$$

and the matrices A and B are given by:

$$A = \begin{bmatrix} (1+\lambda) & -\frac{\lambda}{2} & 0 & 0 \\ -\frac{\lambda}{2} & (1+\lambda) & -\frac{\lambda}{2} & 0 \\ 0 & -\frac{\lambda}{2} & (1+\lambda) & -\frac{\lambda}{2} \\ 0 & 0 & -\frac{\lambda}{2} & (1+\lambda) \end{bmatrix}$$

and

$$B = \begin{bmatrix} (1-\lambda) & \frac{\lambda}{2} & 0 & 0 \\ \frac{\lambda}{2} & (1-\lambda) & \frac{\lambda}{2} & 0 \\ 0 & \frac{\lambda}{2} & (1-\lambda) & \frac{\lambda}{2} \\ 0 & 0 & \frac{\lambda}{2} & (1-\lambda) \end{bmatrix}$$

Then the approximate solution is given by: $\quad w^{(j+1)} = A^{-1}Bw^{(j)}$

PROBLEM DEFINITION

Consider the boundary value problem:

$$\frac{d^2u}{dx^2} = \frac{du}{dt} \qquad 0 < x < 1, \qquad 0 < t < 1$$

$$u(0,t) = 0, \;\; u(1,t) = 0, \qquad 0 \le t \le 1$$

$$u(x,0) = \sin \pi x, \;\; 0 \le x \le 1$$

Use n=5 and m=25.

i. **Finite Difference Method**

$\alpha^2 = 1$

$x0 = 0$

$xL = 1$

$t0 = 0$

$tL = 1$

$n = 5$

$m = 25$

$h = \dfrac{xL - X0}{n} = \dfrac{1}{5} = 0.2$

$k = \dfrac{tL - t0}{m} = \dfrac{1}{25} = 0.04$

$\lambda = \dfrac{\alpha^2 k}{h^2} = \dfrac{1(0.04)}{0.2^2} = 1$

We use formula (6) to solve explicit finite difference:

$$w_{i,j+1} = w_{i+1,j} - w_{i,j} + w_{i-1,j}$$

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

ii. Crank-Nicholson Method

$h = \frac{a}{n}$

$= \frac{1}{5}$

$= 0.2$

$k = \frac{T}{m}$

$= \frac{1}{25}$

$= 0.04$

$\lambda = \frac{ck}{h^2}$

$= \frac{1(0.04)}{(0.2)^2}$

$= 1$

By using formula:

$$A = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & 1 \\ 0 & 0 & -1 & 4 \end{bmatrix}$$

RESULT AND SOLVING

Matlab Coding for solving using Finite-Difference Method:

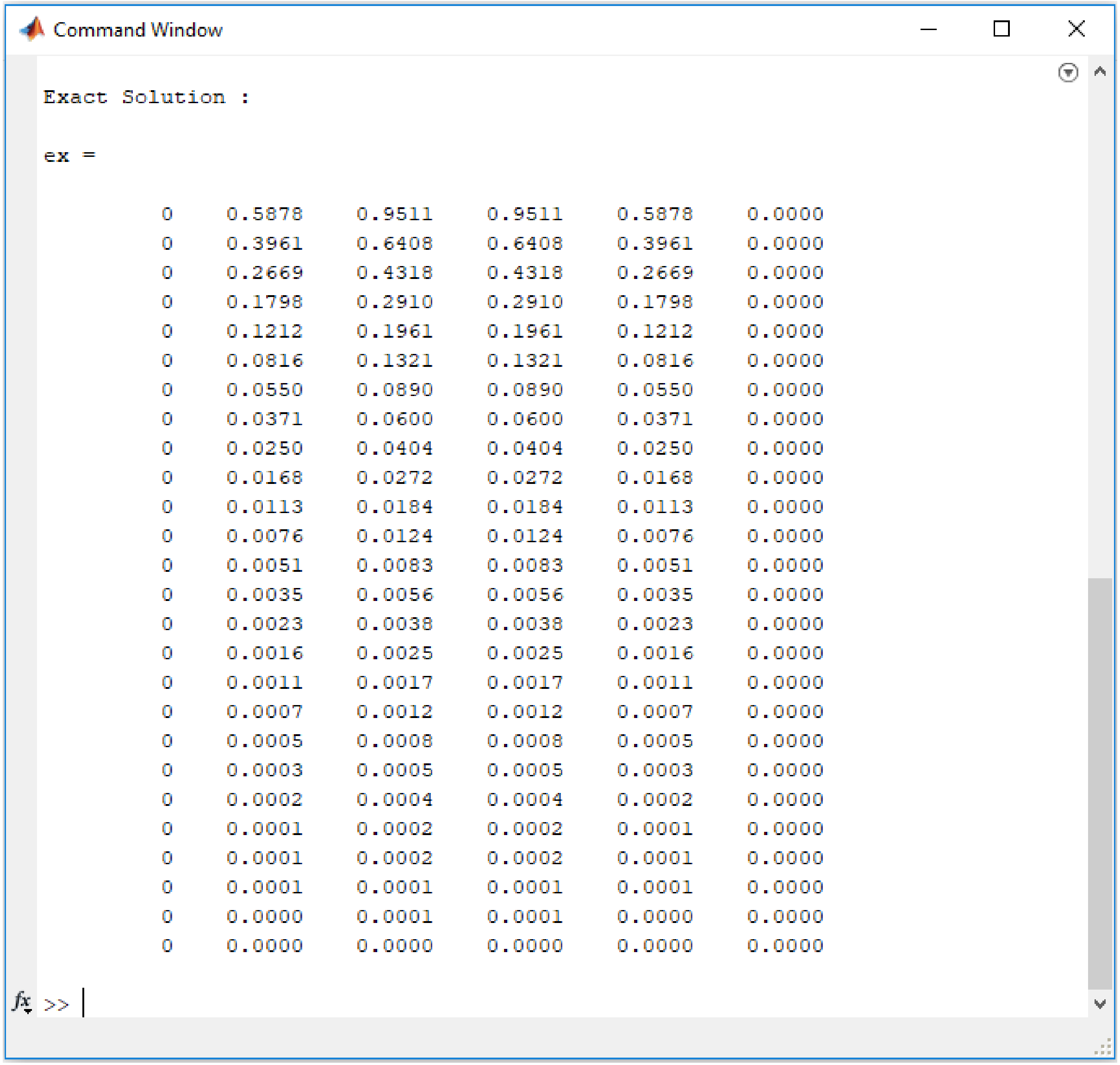
```
function FDMHeat
clc
xO=0;
xL=1;
tO=0;
tL=1;
%number of subinterval
n=4;      %n - mesh for x (in matrices for column,j)
m=24;       %m - mesh for t (in matrices for row,i )
dx=(xL-xO)/(n+1);
dt=(tL-tO)/(m+1);
%alpha^2
a2=1;
r=(a2*dt)/dx^2;
%Matrix A
%main diagonal
for i=1:n
    A(i,i)=(1-2*r);
end

for i=1:n-1
    %to construct tridiagonal
    A(i,i+1)= r;
    A(i+1,i)=r;
end
fprintf('Matrix A:\n');A
%set up BC's (INSERT EQUATION FOR BOUNDARY EQUATIONS)
x=linspace(xO,xL,n+2);
t=linspace(tO,tL,m+2);
tle=@(x,t)0; %U(0,t)=0
tr=@(x,t)0; %U(1,t)=0
tso=@(x,t)sin(pi*x); %U(x,0)=sin(pi*x)

b = zeros(n,1); %U


 for j=1:n
 b(j,1) = tso(x(1,j+1),t(1,1));
 end
 u=[];
 u=b;
for j=1:m+1
 u(:,j+1)=A*u(:,j);
end
u=u.';
```

```
%%
%Matrix P
%to make it easier to plot graph by inserting all the initial condition and
%all the value of u.
p = zeros(m+2,n+2);
for i=1:m+2
    p(i,1) = tle(x(1,1),t(1,i));
    p(i,n+2) = tr(x(1,n+2),t(1,i));
end
for i=1:m+2
    for j=1:n
    p(i,j+1) = u(i,j);
    end
end
p
%%
%Exact Solution
%u(x,t)=exp(-pi*pi*t)*sin(pi*x)
xe=linspace(x0,xL,100);
%for plotting graph
for i=1:100
    for j=1:m+2
        v(i,j)=exp(-pi*pi*t(1,j))*sin(pi*xe(1,i));
    end
end
v=v.';
%to make it comparison between numerical and actual solution (p vs ex)
for i=1:n+2
    for j=1:m+2
        ex(i,j)=exp(-pi*pi*t(1,j))*sin(pi*x(1,i));
    end
end
ex=ex.'
%%
%Graph
 plot(x,p)%numerical solution
 hold on
 plot(xe,v)%exact solution
```

Output for Finite Difference Method:

```
Command Window

Matrix A:

A =

   -1.0000    1.0000         0         0
    1.0000   -1.0000    1.0000         0
         0    1.0000   -1.0000    1.0000
         0         0    1.0000   -1.0000

Results :

p =

         0    0.5878    0.9511    0.9511    0.5878         0
         0    0.3633    0.5878    0.5878    0.3633         0
         0    0.2245    0.3633    0.3633    0.2245         0
         0    0.1388    0.2245    0.2245    0.1388         0
         0    0.0858    0.1388    0.1388    0.0858         0
         0    0.0530    0.0858    0.0858    0.0530         0
         0    0.0328    0.0530    0.0530    0.0328         0
         0    0.0202    0.0328    0.0328    0.0202         0
         0    0.0125    0.0202    0.0202    0.0125         0
         0    0.0077    0.0125    0.0125    0.0077         0
         0    0.0048    0.0077    0.0077    0.0048         0
         0    0.0030    0.0048    0.0048    0.0030         0
         0    0.0018    0.0030    0.0030    0.0018         0
         0    0.0011    0.0018    0.0018    0.0011         0
         0    0.0007    0.0011    0.0011    0.0007         0
         0    0.0004    0.0007    0.0007    0.0004         0
         0    0.0003    0.0004    0.0004    0.0003         0
         0    0.0002    0.0003    0.0003    0.0002         0
         0    0.0001    0.0002    0.0002    0.0001         0
         0    0.0001    0.0001    0.0001    0.0001         0
         0    0.0000    0.0001    0.0001    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000         0
```

Figure 1: Matrix A and Results using Finite-Difference Method



```
Exact Solution :

ex =

         0    0.5878    0.9511    0.9511    0.5878    0.0000
         0    0.3961    0.6408    0.6408    0.3961    0.0000
         0    0.2669    0.4318    0.4318    0.2669    0.0000
         0    0.1798    0.2910    0.2910    0.1798    0.0000
         0    0.1212    0.1961    0.1961    0.1212    0.0000
         0    0.0816    0.1321    0.1321    0.0816    0.0000
         0    0.0550    0.0890    0.0890    0.0550    0.0000
         0    0.0371    0.0600    0.0600    0.0371    0.0000
         0    0.0250    0.0404    0.0404    0.0250    0.0000
         0    0.0168    0.0272    0.0272    0.0168    0.0000
         0    0.0113    0.0184    0.0184    0.0113    0.0000
         0    0.0076    0.0124    0.0124    0.0076    0.0000
         0    0.0051    0.0083    0.0083    0.0051    0.0000
         0    0.0035    0.0056    0.0056    0.0035    0.0000
         0    0.0023    0.0038    0.0038    0.0023    0.0000
         0    0.0016    0.0025    0.0025    0.0016    0.0000
         0    0.0011    0.0017    0.0017    0.0011    0.0000
         0    0.0007    0.0012    0.0012    0.0007    0.0000
         0    0.0005    0.0008    0.0008    0.0005    0.0000
         0    0.0003    0.0005    0.0005    0.0003    0.0000
         0    0.0002    0.0004    0.0004    0.0002    0.0000
         0    0.0001    0.0002    0.0002    0.0001    0.0000
         0    0.0001    0.0002    0.0002    0.0001    0.0000
         0    0.0001    0.0001    0.0001    0.0001    0.0000
         0    0.0000    0.0001    0.0001    0.0000    0.0000
         0    0.0000    0.0000    0.0000    0.0000    0.0000

>>
```

Figure 2: Exact Solution output

Graph:

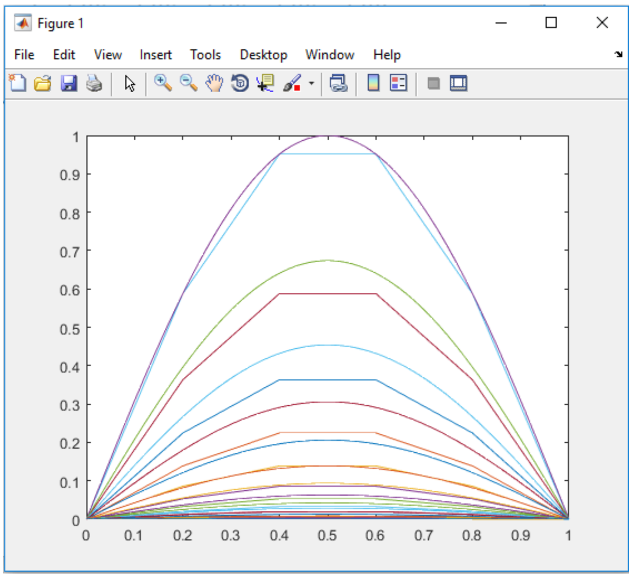


Figure 3: Graph for Finite-Difference Method

Matlab Coding for solving using Crank-Nicholson Method:

```
function CNMHeat
clc
xO=0;
xL=1;
tO=0;
tL=1;
%number of subinterval
n=4;        %n - mesh for x (in matrices for column,j)
m=24;         %m - mesh for y (in matrices for row,i )
dx=(xL-xO)/(n+1);
dt=(tL-tO)/(m+1);
%alpha^2
a2=1;
r=(a2*dt)/dx^2;
%%
%Matrix A

%to construct  main diagonal of matrix
for i=1:n
    A(i,i)=2*(1+r);
end
for i=1:n-1
    %to construct tridiagonal
    A(i,i+1)= -r;
    A(i+1,i)=-r;
end
A
%%
%Matrix B

%to construct diagonal of matrix
for i=1:n
    B(i,i)=2*(1-r);
end
for i=1:n-1
    %to construct tridiagonal
    B(i,i+1)= r;
    B(i+1,i)=r;
end
B

%%
%set up BC's (INSERT EQUATION FOR BOUNDARY EQUATIONS)

x=linspace(xO,xL,n+2);
t=linspace(tO,tL,m+2);
%U(0,t)=0
tle=@(x,t)0;
%U(1,t)=0
tr=@(x,t)0;
%U(x,0)=sin(pi*x)
tso=@(x,t)sin(pi*x);
%%
%U
 b = zeros(n,1);

 for j=1:n
 b(j,1) = tso(x(1,j+1),t(1,1));
```

```
 end
 u=[];
 u=b
for j=1:m+1
 u(:,j+1)=inv(A)*B*u(:,j);
end
u=u.'
%%
%Matrix P(numerical solution)
%to make it easier to plot graph
p = zeros(m+2,n+2);
for i=1:m+2
    p(i,1) = tle(x(1,1),t(1,i));
    p(i,n+2) = tr(x(1,n+2),t(1,i));
end
for i=1:m+2
    for j=1:n
    p(i,j+1) = u(i,j);
    end
end
p
%%
%Exact Solution
%u(x,t)=exp(-pi*pi*t)*sin(pi*x)
xe=linspace(xO,xL,100);
%for plotting graph
for i=1:100
    for j=1:m+2
        v(i,j)=exp(-pi*pi*t(1,j))*sin(pi*xe(1,i));
    end
end
v=v.';
%to make it comparison between numerical and actual solution (p vs ex)
for i=1:n+2
    for j=1:m+2
        ex(i,j)=exp(-pi*pi*t(1,j))*sin(pi*x(1,i));
    end
end
ex=ex.'
%%
%Graph
 plot(x,p)%numerical solution
 hold on
 plot(xe,v)%exact solution
```

Output for Crank-Nicholson Method:

```
Command Window

A =

    4.0000   -1.0000         0         0
   -1.0000    4.0000   -1.0000         0
         0   -1.0000    4.0000   -1.0000
         0         0   -1.0000    4.0000


B =

    0.0000    1.0000         0         0
    1.0000    0.0000    1.0000         0
         0    1.0000    0.0000    1.0000
         0         0    1.0000    0.0000
```

Figure 4: Matrix A and Matrix B using Crank-Nicholson Method

```
Command Window
p =

         0    0.5878    0.9511    0.9511    0.5878         0
         0    0.3993    0.6460    0.6460    0.3993         0
         0    0.2712    0.4388    0.4388    0.2712         0
         0    0.1842    0.2981    0.2981    0.1842         0
         0    0.1251    0.2025    0.2025    0.1251         0
         0    0.0850    0.1376    0.1376    0.0850         0
         0    0.0577    0.0934    0.0934    0.0577         0
         0    0.0392    0.0635    0.0635    0.0392         0
         0    0.0266    0.0431    0.0431    0.0266         0
         0    0.0181    0.0293    0.0293    0.0181         0
         0    0.0123    0.0199    0.0199    0.0123         0
         0    0.0084    0.0135    0.0135    0.0084         0
         0    0.0057    0.0092    0.0092    0.0057         0
         0    0.0039    0.0062    0.0062    0.0039         0
         0    0.0026    0.0042    0.0042    0.0026         0
         0    0.0018    0.0029    0.0029    0.0018         0
         0    0.0012    0.0020    0.0020    0.0012         0
         0    0.0008    0.0013    0.0013    0.0008         0
         0    0.0006    0.0009    0.0009    0.0006         0
         0    0.0004    0.0006    0.0006    0.0004         0
         0    0.0003    0.0004    0.0004    0.0003         0
         0    0.0002    0.0003    0.0003    0.0002         0
         0    0.0001    0.0002    0.0002    0.0001         0
         0    0.0001    0.0001    0.0001    0.0001         0
         0    0.0001    0.0001    0.0001    0.0001         0
         0    0.0000    0.0001    0.0001    0.0000         0
```

Figure 5: Results using Crank-Nicholson Method

```
Command Window

ex =

         0    0.5878    0.9511    0.9511    0.5878    0.0000
         0    0.3961    0.6408    0.6408    0.3961    0.0000
         0    0.2669    0.4318    0.4318    0.2669    0.0000
         0    0.1798    0.2910    0.2910    0.1798    0.0000
         0    0.1212    0.1961    0.1961    0.1212    0.0000
         0    0.0816    0.1321    0.1321    0.0816    0.0000
         0    0.0550    0.0890    0.0890    0.0550    0.0000
         0    0.0371    0.0600    0.0600    0.0371    0.0000
         0    0.0250    0.0404    0.0404    0.0250    0.0000
         0    0.0168    0.0272    0.0272    0.0168    0.0000
         0    0.0113    0.0184    0.0184    0.0113    0.0000
         0    0.0076    0.0124    0.0124    0.0076    0.0000
         0    0.0051    0.0083    0.0083    0.0051    0.0000
         0    0.0035    0.0056    0.0056    0.0035    0.0000
         0    0.0023    0.0038    0.0038    0.0023    0.0000
         0    0.0016    0.0025    0.0025    0.0016    0.0000
         0    0.0011    0.0017    0.0017    0.0011    0.0000
         0    0.0007    0.0012    0.0012    0.0007    0.0000
         0    0.0005    0.0008    0.0008    0.0005    0.0000
         0    0.0003    0.0005    0.0005    0.0003    0.0000
         0    0.0002    0.0004    0.0004    0.0002    0.0000
         0    0.0001    0.0002    0.0002    0.0001    0.0000
         0    0.0001    0.0002    0.0002    0.0001    0.0000
         0    0.0001    0.0001    0.0001    0.0001    0.0000
         0    0.0000    0.0001    0.0001    0.0000    0.0000
         0    0.0000    0.0000    0.0000    0.0000    0.0000

fx >>
```

Figure 6: Exact Solution output

Graph:

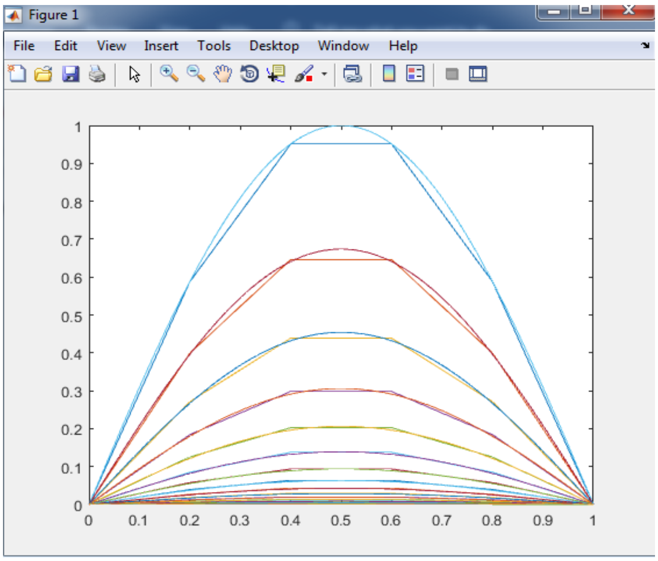


Figure 7: Graph for Crank-Nicholson Method

CONCLUSION

Table 1: Comparing actual value with numerical value of Finite Difference Method

x_i	$w(x_i, 0.40)$	FDM $u(x_i, 0.40)$	$\lvert u(x_i, 0.40)- w(x_i, 0.40)\rvert$
0	0	0	
0.20	0.0113	0.0048	6.5×10^{-3}
0.40	0.0184	0.0077	1.07×10^{-2}
0.60	0.0184	0.0077	6.5×10^{-3}
0.8	0.0113	0.0048	1.07×10^{-2}
1	0	0	

Table 2: Comparing actual value with numerical value of Crank-Nicolson Method

x_i	$w(x_i, 0.40)$	CNM $u(x_i, 0.40)$	$\lvert u(x_i, 0.40)- w(x_i, 0.40)\rvert$
0	0	0	
0.20	0.0113	0.0123	1×10^{-3}
0.40	0.0184	0.0199	1.5×10^{-3}
0.60	0.0184	0.0199	1.5×10^{-3}
0.8	0.0113	0.0123	1×10^{-3}
1	0	0	

Table 1 and Table 2 show the result when t=0.40 by comparing the numerical solution with analytical solution. As a conclusion, by comparing the results of Finite-Difference Method and Crank-Nicholson Method, it is clearly shown that the Crank-Nicholson Method is more accurate and has less error compared to the Finite Difference Method.

In Figure 3 shows that the graph of Finite Difference Method is far away from actual solution while Figure 7 shows that the graph of Crank Nicolson Method is approaching towards the actual solution. However, the graph shown for numerical solution is not accurately as actual solution, this is due to small value of n which is 5. Hence, we need to increase the value of n, so that the step size for x is smaller. The smaller the step size of x, the better approximation to the actual solution.

REFERENCES

Nicotson, E. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, *6*(2), 207–208.